Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

# Системне програмування

# Лабораторна робота №4

«Програмування арифметичних операцій підвищеної розрядності»

Виконав:
студент групи ІО-82
Шендріков Є. О.
Залікова № 8227
Перевірив Порєв В. М.

Київ - 2020 р.

## **Мета**:

Навчитися програмувати на асемблері основні арифметичні операції підвищеної розрядності, а також отримати перші навички програмування власних процедур у модульному проекті.

## **Завдання:**

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab4.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути три модуля на асемблері: - головний модуль: файл main4.asm. Цей модуль створити та написати заново, частково використавши текст модуля main3.asm попередньої роботи №3; - другий модуль: використати module попередньої роботи №3; - третій модуль: створити новий з ім'ям longop.

3. У цьому проекті кожний модуль може окремо компілюватися.

4. Скомпілювати вихідний текст і отримати виконуємий файл програми.

5. Перевірити роботу програми. Налагодити програму.

6. Отримати результати – кодовані значення чисел згідно варіанту завдання.

7. Проаналізувати та прокоментувати результати, вихідний текст та дизасемблерний машинний код програми.

## **Текст програми:**

*main4.asm*

```
.586
.model flat, c

include \masm32\include\user32.inc
include \masm32\include\kernel32.inc

include module.inc
include longop.inc

includelib \masm32\lib\user32.lib              ;Включення деяких бібліотек
includelib \masm32\lib\kernel32.lib
```

```
.const                                          ;Ініціалізація const параметрів
Caption db "Assembler - Lab4", 0


.data                                           ;Ініціалізація global параметрів
Caption1 db "A+B 1",0
Caption3 db "A+B 2",0
Caption2 db "A-B",0

TextBuf1 db 864 dup(?)
TextBuf3 db 864 dup(?)
TextBuf2 db 256 dup(?)

ValueA1 dd 864 dup(?)
ValueB1 dd 864 dup(?)

ValueA3 dd 19h, 1Ah, 1Bh, 1Ch, 1Dh, 1Eh, 1Fh, 20h, 21h, 22h, 23h, 24h, 25h, 26h, 27h,
28h, 29h, 2Ah, 2Bh, 2Ch, 2Dh, 2Eh, 2Fh, 30h, 31h, 32h, 33h
ValueB3 dd 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h,
80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h,
80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h, 80000001h,
80000001h,  80000001h, 80000001h, 80000001h

ValueA2 dd 256 dup(0)
ValueB2 dd 25h, 26h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch

Result1 dd 864 dup(0)
Result3 dd 864 dup(0)
Result2 dd 256 dup(0)

.code
main:
            ;A+B 1
            mov eax, 80010001h
            mov ecx, 27 ; ECX = потрібна кількість повторень
            mov edx, 0
cycleAB1:
            mov DWord ptr[ValueA1+4*edx], eax
            mov DWord ptr[ValueB1+4*edx], 80000001h
            add eax, 10000h
            inc edx
            dec ecx ; зменшуємо лічильник на 1
            jnz cycleAB1

            push offset ValueA1
            push offset ValueB1
            push offset Result1
            call Add_864_LONGOP
            push offset TextBuf1
            push offset Result1
            push 864
            call StrHex_MY
            invoke MessageBoxA, 0, ADDR TextBuf1, ADDR Caption1,0

            ;A+B 2
            push offset ValueA3
            push offset ValueB3
            push offset Result3
```

```
            call Add_864_LONGOP
            push offset TextBuf3
            push offset Result3
            push 864
            call StrHex_MY
            invoke MessageBoxA, 0, ADDR TextBuf3, ADDR Caption3,0

            ;A-B
            push offset ValueA2
            push offset ValueB2
            push offset Result2
            call Sub_256_LONGOP
            push offset TextBuf2
            push offset Result2
            push 256
            call StrHex_MY
            invoke MessageBoxA, 0, ADDR TextBuf2, ADDR Caption2,0
            invoke ExitProcess, 0
end main
```

*longop.asm*

```
.586
.model flat, c
.code

Add_864_LONGOP proc
    push ebp
        mov ebp,esp

        mov esi, [ebp+16] ; ESI = адреса A
        mov ebx, [ebp+12] ; EBX = адреса B
        mov edi, [ebp+8] ; EDI = адреса результату

        mov ecx, 27; 864/32 ECX = потрібна кількість повторень
        mov edx, 0
        clc ; обнулює біт CF регістру
        cycle:
        mov eax, dword ptr[esi+4*edx]
        adc eax, dword ptr[ebx+4*edx] ; додавання групи з 32 бітів
        mov dword ptr[edi+4*edx], eax

        inc edx
        dec ecx ; лічильник зменшуємо на 1
        jnz cycle
        pop ebp
        ret 12
Add_864_LONGOP endp

Sub_256_LONGOP proc
    push ebp
        mov ebp,esp

        mov esi, [ebp+16] ; ESI = адреса A
        mov ebx, [ebp+12] ; EBX = адреса B
        mov edi, [ebp+8] ; EDI = адреса результату

        mov ecx, 8; 256/32 ECX = потрібна кількість повторень
```

```
        mov edx, 0
        clc ; обнулює біт CF регістру
        cycle:
        mov eax, dword ptr[esi+4*edx]
        sbb eax, dword ptr[ebx+4*edx] ; віднімання групи з 32 бітів
        mov dword ptr[edi+4*edx], eax

        inc edx
        dec ecx ; лічильник зменшуємо на 1
        jnz cycle
        pop ebp
        ret 12
Sub_256_LONGOP endp

End
```

*module.asm*

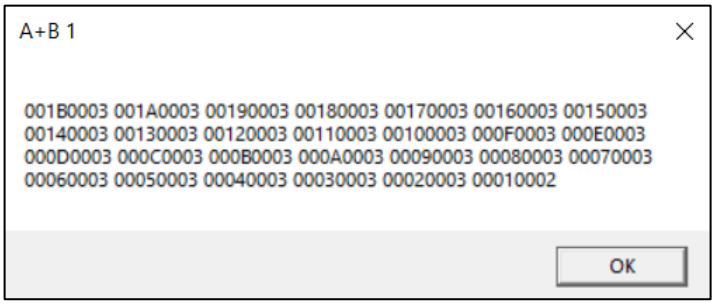
```
.586
.model flat, c
.code

;процедура StrHex_MY записує текст шістнадцятькового коду
;перший параметр - адреса буфера результату (рядка символів)
;другий параметр - адреса числа
;третій параметр - розрядність числа у бітах (має бути кратна 8)
StrHex_MY proc
       push ebp
       mov ebp,esp
       mov ecx, [ebp+8]                        ;кількість бітів числа
       cmp ecx, 0
       jle @exitp
       shr ecx, 3                              ;кількість байтів числа
       mov esi, [ebp+12]                       адреса числа
       mov ebx, [ebp+16]                       ;адреса буфера результату
@cycle:
       mov dl, byte ptr[esi+ecx-1]             ;байт числа - це дві hex-цифри
       mov al, dl
       shr al, 4                               ;старша цифра
       call HexSymbol_MY
       mov byte ptr[ebx], al
       mov al, dl                              ;молодша цифра
       call HexSymbol_MY
       mov byte ptr[ebx+1], al
       mov eax, ecx
       cmp eax, 4
       jle @next
       dec eax
       and eax, 3                              проміжок розділює групи по вісім цифр
       cmp al, 0
       jne @next
       mov byte ptr[ebx+2], 32                 ;код символа проміжку
       inc ebx

@next:
       add ebx, 2
       dec ecx
       jnz @cycle
       mov byte ptr[ebx], 0                    ;рядок закінчується нулем
```

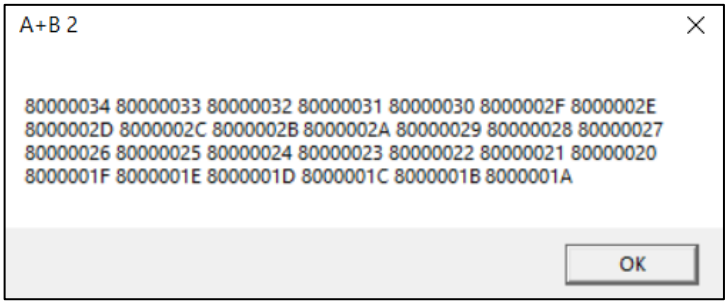
```
@exitp:
	pop ebp
	ret 12

StrHex_MY endp


;ця процедура обчислює код hex-цифри
;параметр - значення AL
;результат -> AL
HexSymbol_MY proc
	and al, 0Fh
	add al, 48				;так можна тільки для цифр 0-9
	cmp al, 58
	jl @exitp
	add al, 7				;для цифр A,B,C,D,E,F
@exitp:
	ret
HexSymbol_MY endp

DwordToStrHex proc
	push ebp
	mov ebp,esp
	mov ebx,[ebp+8]				;другий параметр
	mov edx,[ebp+12]			;перший параметр
	xor eax,eax
	mov edi,7

@next:
	mov al,dl
	and al,0Fh				;виділяємо одну шістнадцяткову цифру
	add ax,48				;так можна тільки для цифр 0-9
	cmp ax,58
	jl @store
	add ax,7				;для цифр A,B,C,D,E,F
@store:
	mov [ebx+edi],al
	shr edx,4
	dec edi
	cmp edi,0
	jge @next
	pop ebp
	ret 8
DwordToStrHex endp

end
```

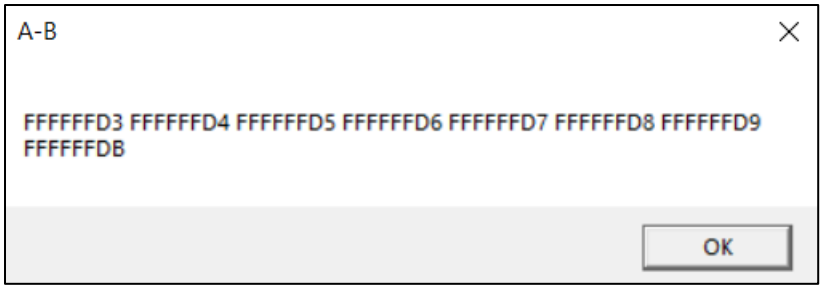
## Аналіз результатів:

Дана програма виконує операції віднімання і додавання з числами підвищеної розрядності.

## Результати роботи програми:




## Висновок:

Навчився програмувати на асемблері операції додавання і віднімання чисел з підвищеною розрядностю, а також отримав перші навички програмування власних процедур у модульному проекті. Кінцева мета роботи досягнута.